algorithms

MDPI

Article

Convolutional-Neural-Network-Based Handwritten Character Recognition: An Approach with Massive Multisource Data

Nazmus Saqib [1,*], Khandaker Foysal Haque [2], Venkata Prasanth Yanambaka [1] and Ahmed Abdelgawad [1]

[1] College of Science and Engineering, Central Michigan University, Mount Pleasant, MI 48859, USA; yanam1v@cmich.edu (V.P.Y.); abdel1a@cmich.edu (A.A.)
[2] Institute for the Wireless Internet of Things, Northeastern University, Boston, MA 02115, USA; haque.k@northeastern.edu
* Correspondence: saqib1n@cmich.edu

Abstract: Neural networks have made big strides in image classification. Convolutional neural networks (CNN) work successfully to run neural networks on direct images. Handwritten character recognition (HCR) is now a very powerful tool to detect traffic signals, translate language, and extract information from documents, etc. Although handwritten character recognition technology is in use in the industry, present accuracy is not outstanding, which compromises both performance and usability. Thus, the character recognition technologies in use are still not very reliable and need further improvement to be extensively deployed for serious and reliable tasks. On this account, characters of the English alphabet and digit recognition are performed by proposing a custom-tailored CNN model with two different datasets of handwritten images, i.e., Kaggle and MNIST, respectively, which are lightweight but achieve higher accuracies than state-of-the-art models. The best two models from the total of twelve designed are proposed by altering hyper-parameters to observe which models provide the best accuracy for which dataset. In addition, the classification reports (CRs) of these two proposed models are extensively investigated considering the performance matrices, such as precision, recall, specificity, and *F*1 score, which are obtained from the developed confusion matrix (CM). To simulate a practical scenario, the dataset is kept unbalanced and three more averages for the F measurement (micro, macro, and weighted) are calculated, which facilitates better understanding of the performances of the models. The highest accuracy of 99.642% is achieved for digit recognition, with the model using 'RMSprop', at a learning rate of 0.001, whereas the highest detection accuracy for alphabet recognition is 99.563%, which is obtained with the proposed model using 'ADAM' optimizer at a learning rate of 0.00001. The macro *F*1 and weighted *F*1 scores for the best two models are 0.998, 0.997:0.992, and 0.996, respectively, for digit and alphabet recognition.

Keywords: handwritten character recognition; English character recognition; convolutional neural networks (CNNs); deep learning in character recognition; digit recognition; English alphabet recognition

Citation: Saqib, N.; Haque, K.F.; Yanambaka, V.P.; Abdelgawad, A. Convolutional-Neural-Network-Based Handwritten Character Recognition: An Approach with Massive Multisource Data. *Algorithms* **2022**, *15*, 129. https://doi.org/10.3390/a15040129

Academic Editor: Marcos Zampieri

Received: 5 March 2022
Accepted: 12 April 2022
Published: 14 April 2022

1. Introduction

Handwriting is the most typical and systematic way of recording facts and information. The handwriting of an individual is idiosyncratic and unique to individual people. The capability of software or a device to recognize and analyze human handwriting in any language is called a handwritten character recognition (HCR) system. Recognition can be performed from both online and offline handwriting. In recent years, applications of handwriting recognition are thriving, widely used in reading postal addresses, language translation, bank forms and check amounts, digital libraries, keyword spotting, and traffic sign detection.

Image acquisition, preprocessing, segmentation, feature extraction, and classification are the typical processes of an HCR system, as shown in Figure 1. The initial step is to receive an image form of handwritten characters, which is recognized as image acquisition

that will proceed as an input to preprocessing. In preprocessing, distortions of the scanned images are removed and converted into binary images. Afterward, in the segmentation step, each character is divided into sub images. Then, it will extract every characteristic of the features from each image of the character. This stage is especially important for the last step of the HCR system, which is called classification [1]. Based on classification accuracy and different approaches to recognize the images, there are many classification methods, i.e., convolutional neural networks (CNNs), support vector machines (SVMs), recurrent neural networks (RNNs), deep belief networks, deep Boltzmann machines, and K-nearest neighbor (KNN) [2].

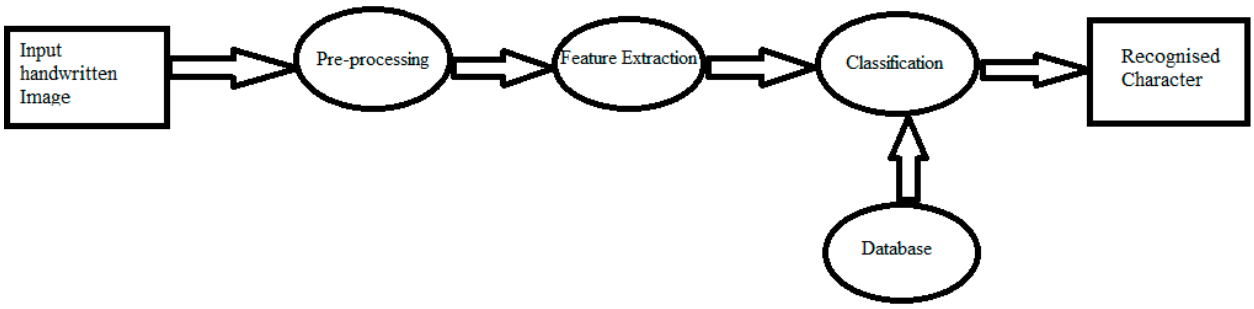

Figure 1. Representation of a common handwritten character recognition (HCR) system.

A subclass of machine learning comprises neural networks (NNs), which are information-processing methods inspired by the biological process of the human brain. Figure 2 represents the basic neural network. The number of layers is indicated by deep learning in a neural network. Neurons, being the information-processing element, build the foundation of neural networks that draws parallels from the biological neural network. Weights associated with the connection links, bias, inputs, and outputs are the primary components of an NN. Every node is called a perceptron in a neural network (NN) [3]. Research is being conducted to obtain the best accuracy, but the accuracy using a CNN is not outstanding, which compromises the performance and usability for handwritten character recognition. Hence, the aim of this paper is to obtain the highest accuracy by introducing a handwritten character recognition (HCR) system using a CNN, which can automatically extract the important features from the images better than multilayer perceptron (MLP) [4–9].

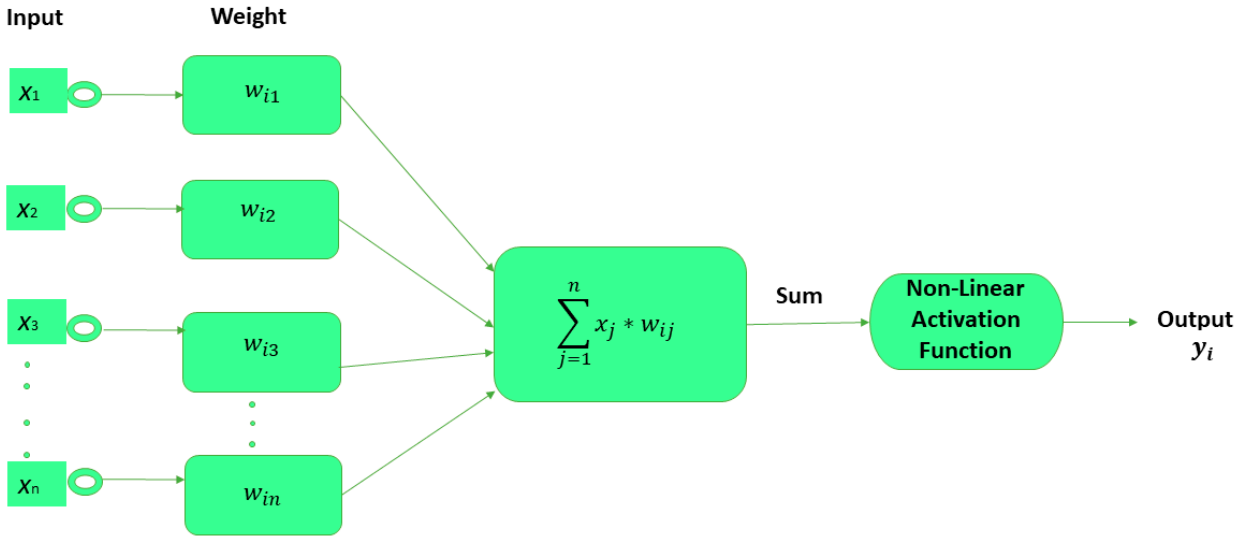

Figure 2. Representation of a basic neural network (NN).

CNNs were first employed in 1980 [10]. The conception of convolutional neural networks (CNNs) was motivated by the human brain. People can identify objects from their childhood because they have seen hundreds of pictures of those objects, which is why a child can guess an object that they have never seen before. CNNs work in a similar way. CNNs used for analyzing visual images are a variation of an MLP deep neural network that is fully connected. Fully connected means that each neuron in the layer is fully connected to all the neurons in the subsequent layer. Some of the renowned CNN architectures are AlexNet (8 layers), VGG (16, 19 layers), GoogLeNet (22 layers), and ResNet (152 layers) [11]. CNN models can provide an excellent recognition result because they do not need to collect prior knowledge of designer features. As for CNNs, they do not depend on the rotation of input images.

A CNN model has been broadly set for the HCR system, using the MNIST dataset. Such research has been carried out for several years. A few researchers have found the accuracy to be up to 99% for the recognition of handwritten digits [12]. An experiment was carried out using a combination of multiple CNN models for MNIST digits and had 99.73% accuracy [13]. Afterward, for the same MNIST dataset, the recognition accuracy was improved to 99.77%, when this experiment of the 7-net committee was extended to a 35-net committee [14]. Niu and Suen minimized the structural risk by integrating the SVM for the MNIST digit recognition and obtain the astonishing accuracy of 99.81% [15]. Chinese handwritten character recognition was investigated using a CNN [16]. Recently, Alvear-Sandoval et al. worked on deep neural networks (DNN) for MNIST and obtained a 0.19% error rate [17]. Nevertheless, after a vigilant investigation, it has been observed that the maximal recognition accuracy of the MNIST dataset can be attained by using only ensemble methods, as these aid in improving the classification accuracy. However, there are tradeoffs, i.e., high computational cost and increased testing complexity [18]. In this paper, a tailored CNN model is proposed which attains higher accuracy with light computational complexity.

Research on HCR technology has been going on for long time now and it is in use by the industry, but the accuracy is low, which compromises the usability and overall performance of the technology. Until now, the character recognition technologies in use are still not very dependable and need more development to be deployed broadly for unfailing applications. On this account, characters of the English alphabet and digit recognition are performed in this paper by proposing a custom-tailored CNN model with two different datasets of handwritten images, i.e., Kaggle and MNIST, respectively, which achieve higher accuracies. The important features of these proposed projects are as follows:

1. In the proposed CNN model, four 2D convolutional layers are kept the same and unchanged to obtain the maximum comparable recognition accuracy into two different datasets, Kaggle and MNIST, for handwritten letters and digits, respectively. This proves the versatility of our proposed model.
2. A custom-tailored, lightweight, high-accuracy CNN model (with four convolutional layers, three max-pooling layers, and two dense layers) is proposed by keeping in mind that it should not overfit. Thus, the computational complexity of our model is reduced.
3. Two different optimizers are used for each of the datasets, and three different learning rates (LRs) are used for each of the optimizers to evaluate the best models of the twelve models designed. This suitable selection will assist the research community in obtaining a deeper understanding of HCR.
4. To the best of the authors' knowledge, the novelty of this work is that no researchers to date have worked with the classification report in such detail with a tailored CNN model generalized for both handwritten English alphabet and digit recognition. Moreover, the proposed CNN model gives above 99% recognition accuracy both in compact MNIST digit datasets and in extensive Kaggle datasets for alphabets.
5. The distribution of the dataset is imbalanced. Hence, only the accuracy would be ineffectual in evaluating model performance, so advanced performances are analyzed

to a great extent with a classification report for the best two proposed models for the Kaggle and MNIST datasets, respectively. Classification reports indicate the *F*1 score for each of the 10 classes for digits (0–9) and each of the 26 classes for alphabet (A–Z). In our case of multiclass classification, we examined averaging methods for the *F*1 score, resulting in different average scores, i.e., micro, macro, and weighted average, which is another novelty of this proposed project.

The rest of the paper is organized as follows: Section 2 describes the review of the literature and related works in the handwritten character recognition research arena; Sections 3 and 4 present datasets and proposed CNN model architecture, respectively; Section 5 discusses the result analysis and provides a comparative analysis; and Section 6 describes the conclusion and suggestions for future directions.

2. Review of Literature and Related Works

Many new techniques have been introduced in research papers to classify handwritten characters and numerals or digits. Shallow networks have already shown promising results for handwriting recognition [19–26]. Hinton et al. investigated deep belief networks (DBN), which have three layers along with a grasping algorithm, and recorded an accuracy of 98.75% for the MNIST dataset [27]. Pham et al. improved the performance of recurrent neural networks (RNNs), reducing the word error rate (WER) and character error rate (CER) by employing a regularization method of dropout to recognize unconstrained handwriting [28].

The convolutional neural network (CNN) delivered a vast change as it delivers a state-of-the-art performance in HCR accuracy [29–33]. In 2003, for visual document analysis, a common CNN architecture was introduced by Simard et al., which loosened the training of complex methods of neural networks [34]. Wang et al. used multilayer CNNs for end-to-end text recognition on benchmark datasets, e.g., street view text and ICDAR 2003, and accomplished brilliant results [35].

Recently, for scene text recognition, Shi et al. introduced a new approach, the conventional recurrent neural network (CRNN), integrating both the deep CNN (DCNN) and recurrent neural network (RNN), and announced its superiority to traditional methods of character recognition [36]. For semantic segmentation, Badrinarayanan et al. proposed a deep convolutional network architecture where the max-pooling layer was used to obtain good performance; the authors also compared their model with current techniques. The segmentation architecture known as SegNet consists of a pixel-wise classification layer, an encoder network, and a decoder network [37,38]. In offline handwritten character recognition, CNN has shown outstanding performance for different regional and international languages. Researchers have conducted studies on Chinese handwritten text recognition [39–41]; Arabic language [42]; handwritten Urdu text recognition [43,44]; handwritten Tamil character recognition [45]; Telugu character recognition [46]; and handwritten character recognition on Indic scripts [47].

Gupta et al. used features extracted from a CNN in their model and recognized the informative local regions in [48] from recent character images, accomplishing a recognition accuracy of 95.96% by applying a novel multi-objective optimization framework for HCR which comprises handwritten Bangla numerals, handwritten Devanagari characters, and handwritten English numerals. High performance of the CROHME dataset was observed in the work of Nguyen et al. [49]. The author employed a multiscale CNN for clustering handwritten mathematical expression (HME) and concluded by identifying that their model can be improved by training the CNN with a combination of global, attentive, and max-pooling layers.

Recognition of word location in historical books, for example on Gutenberg's Bible pages, is wisely addressed in the work of Ziran et al. [50] by developing an R-CNN-based deep learning framework. Ptucha et al. introduced an intelligent character recognition (ICR) system, logically using a conventional neural network [51]. IAM datasets and French-language-based RIMES lexicon datasets were used to evaluate the model,

which reported a commendable result. The variance between model parameters and hyper-parameters was highlighted in [52]. The hyper-parameters include the number of epochs, hidden units, hidden layers, learning rate (LR), kernel size, activation function, etc., which must be determined before the training begins to determine the performance of the CNN [53]. It is mentioned that, if the hyper-parameters are chosen poorly, it can lead to a bad CNN performance. The total number of hyper-parameters of some CNN models are 27, 57, 78, and 150, respectively, for AlexNet [54], VGG-16 [55], GoogleNet [56], and ResNet-52 [57]. To improve the recognition performance, practicing researchers play an important role in the handwriting recognition field for designing CNN parameters effectively. Tapotosh Ghosh et al. converted the images into black-and-white 28 × 28 forms with white as the foreground color in [58] by approaching InceptionResNetV2, DenseNet121, and InceptionNetV3 using the CMATERdb dataset. The accuracy obtained by different researchers, their dataset preprocessing, and the different approaches taken to obtain the best recognition accuracy in recent years have been arranged in a tabular form at the end of the paper in Section 5—Results and Analysis.

3. Datasets

The MNIST digit benchmark dataset is a subgroup of a bigger special dataset available from the National Institute of Standards and Technology (NIST). This benchmark dataset, having two categories (digit and alphabet), is accessible through Keras functionality, which is shaped through training on 60,000 sets of examples and a test set, which is made up of testing 10,000 examples [59]. Nevertheless, for digit recognition in this project, only 1 type of dataset is used from the list, which comprises 10 classes of MNIST digits.

Each digit is of uniform size and, by computing the center of mass of the pixels, each binary image of a handwritten digit is centered into a 28 × 28 image. The test set consists of 5000 patterns and each image consists of 30,000 patterns from 2 datasets, from about 250 different writers, 1 from high school students and the other from Census Bureau employees [1]. To make verification easier, datasets are labeled accordingly. The MNIST images sample distribution is shown in Figure 3.

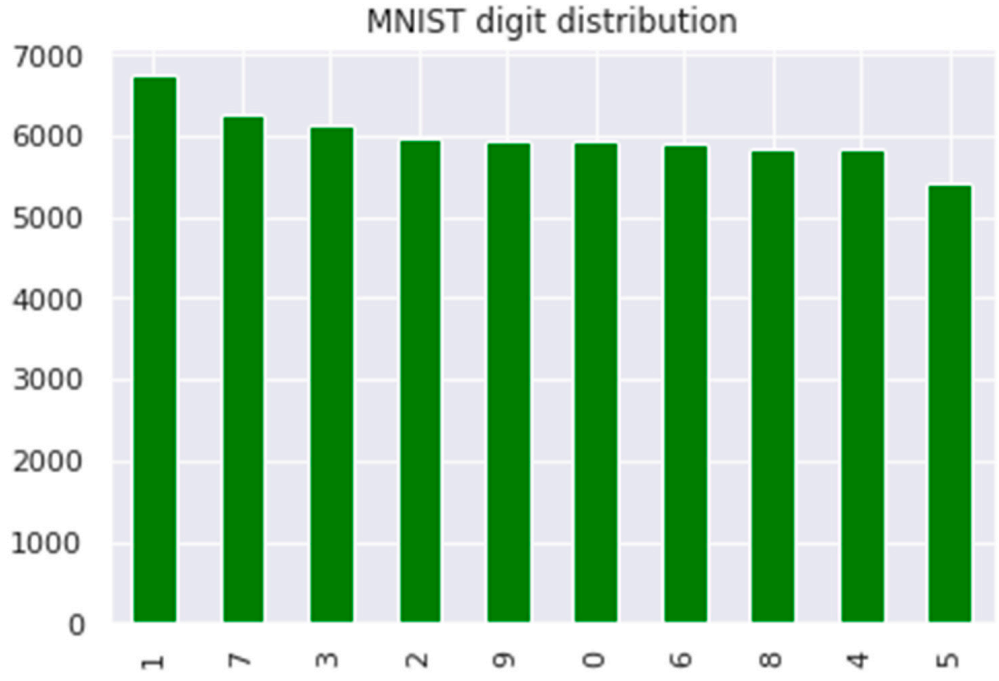

Figure 3. Total distribution of MNIST digits (0–9).

The Kaggle alphabet dataset was sourced from the National Institute of Standards and Technology (NIST), NMIST, and other google images [60]. Kaggle English handwritten alphabets of 26 classes are shaped by training with over 297,000 sets of examples and a test set, which is made up of over 74,490 examples. The total distribution of Kaggle letters is illustrated in Figure 4. Each letter is of uniform size and by computing the center of mass of the pixels, each binary image of a handwritten letter is centered into a 28 × 28 image.

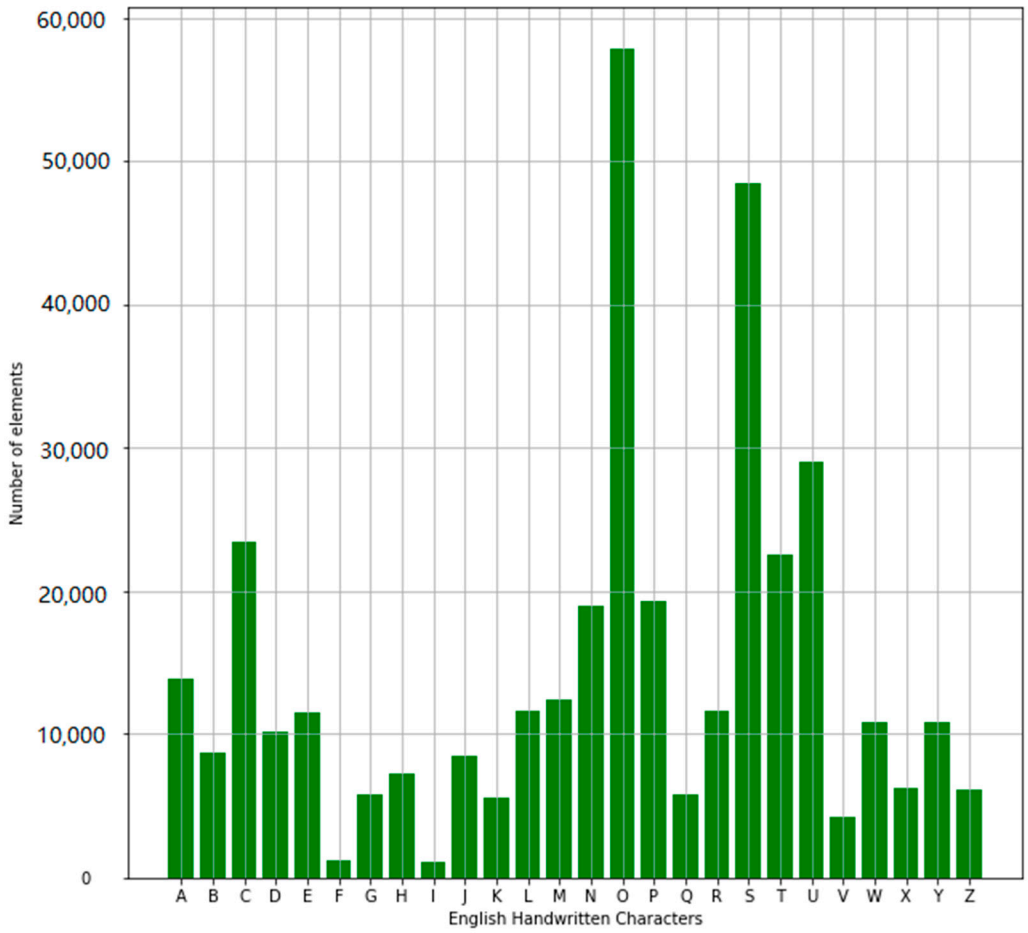

Figure 4. Total distribution of Kaggle letters (A–Z).

4. Proposed Convolutional Neural Network

Of all the deep learning models in image classifications, CNN has become very popular due to its high performance in recognizing image patterns. This has opened up various application opportunities in our daily life and industries which include medical image classification, traffic monitoring, autonomous object recognition, facial recognition, and much more.

CNNs are sparse, feed-forward neural networks [61]. The idea of an artificial neuron was first conceptualized in 1943. Hubel and Wisel first found that, for detecting lights in the receptive fields, visual cortex cells have a major role, which greatly inspired building models such as neocognitron. This model is considered to be the base and predecessor of CNN. CNN is formed of artificial neurons which have a self-optimization property, learning like brain neurons. Due to this self-optimizing property, it can extract and classify the features extracted from images more precisely than any other algorithm. Moreover, it needs very limited preprocessing of the input data, while yielding highly accurate and precise results. CNNs are vastly used in object detection and image classification, including medical imaging. In image classification, each pixel is considered a feature for the neural network. CNN tries to understand and differentiate among the images depending

on these features. Conventionally, first few convolutional layers capture very low-level features, such as the edges, gradient orientation, or color. However, with the increased number of convolutional layers, it starts extracting high-level features. Due to the higher dimensionality and convolution, the parameters of the network increase exponentially. This makes the CNN computationally heavy. However, with the development of computational technology and GPU, these jobs have become much more efficient. Moreover, the development of the CNN algorithms has also prompted the ability to reduce dimensionality by considering small patches at a time which reduces the computational burden without losing the important features.

Handwritten character recognition (HCR) with deep learning and CNN was one of the earliest endeavors of researchers in the field. However, with increased modeling efficacy and the availability of a huge dataset, current models can perform significantly better than the models of ten years ago. However, one of the challenges of the current models is generalization. The model that performs excellently with one dataset may perform poorly with a different one. Thus, it is important to develop a robust model which can perform with the same level of accuracy across different datasets, which would give the model versatility. Thus, a CNN model is designed which is computationally proficient because of its optimized number of CNN layers, while performing with high accuracy across multisource massive datasets.

Owing to the lower resolution of the handwritten character images, the images which were fed to the input layers were sized 28 × 28 pixels. The input layer feeds the images to the convolutional layers, where the features are convolved. The model has only four convolutional layers, which makes it lightweight and computationally efficient. The first layer is a 2D convolutional layer with a 3 × 3 kernel size and rectified linear unit (ReLU)-activation function. ReLU is one of the most widely used activation functions in deep learning algorithms. ReLU is computationally effective because the neurons are not activated altogether like the other activation functions, e.g., tanh [62]. ReLU is a piecewise linear function which is also continuous and differentiable at all points except for 0. Besides providing simplicity and empirical simplicity, it also has reduced likelihood of vanishing gradient. Because of the abovementioned benefits, and as per the suggestion of the literature that ReLUs tend to converge early, it was chosen for our model. The idea behind ReLU is simple, it returns positive values input directly to the output, whereas the negative values are returned as 0, as depicted in Figure 5.

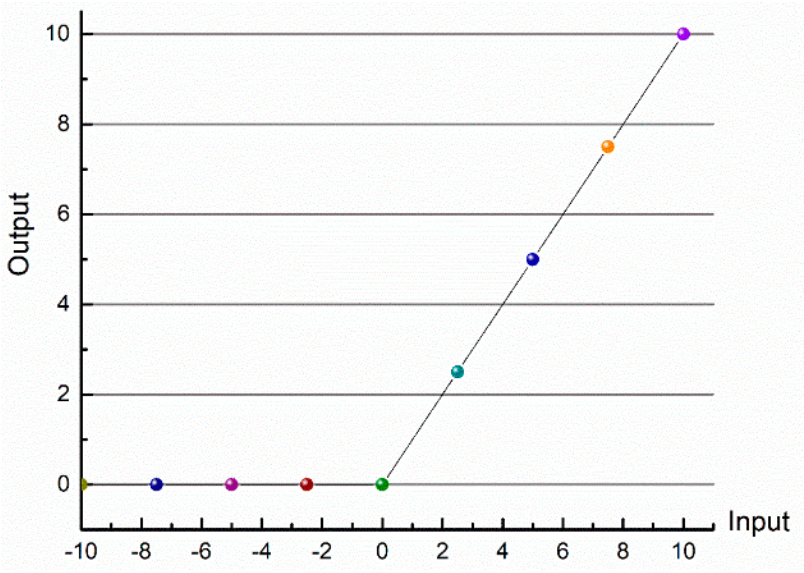

Figure 5. Rectified linear unit (ReLU) function.

The subsequent three layers are the 2D convolutional layers, which are accompanied by one max-pooling layer and a ReLU-activation function. Max pooling is a sample-based discretization process which is used to downsize our input images. It pools the maximum value from each patch of each feature map, thus helping to reduce the dimensionality of the network. Moreover, it reduces the number of parameters by discarding insignificant ones, which decreases the computational burden as well as helping to avoid overfitting. Thus, a 2 × 2 max-pooling layer is integrated in each of the convolutional layers except for the first one. The output of the fourth convolutional layer is fed to the flattening layer to convert the input to a 1D string, which is then fed to the fully connected layer, i.e., the dense layer.

In the fully connected layer, as the name suggests, all the neurons are linked to the activation units of the following layer. In the proposed model, there are two fully connected layers where all the neurons of the first layer are connected to the activation unit of the second fully connected layer. In the second fully connected layer, all the inputs are passed to the Softmax activation function, which categorizes the features into multiclass as needed. Finally, the determined class of any input image is declared in the output. The proposed model is illustrated in Figure 6 and the resultant parameters of each layer are tabulated in Table 1.

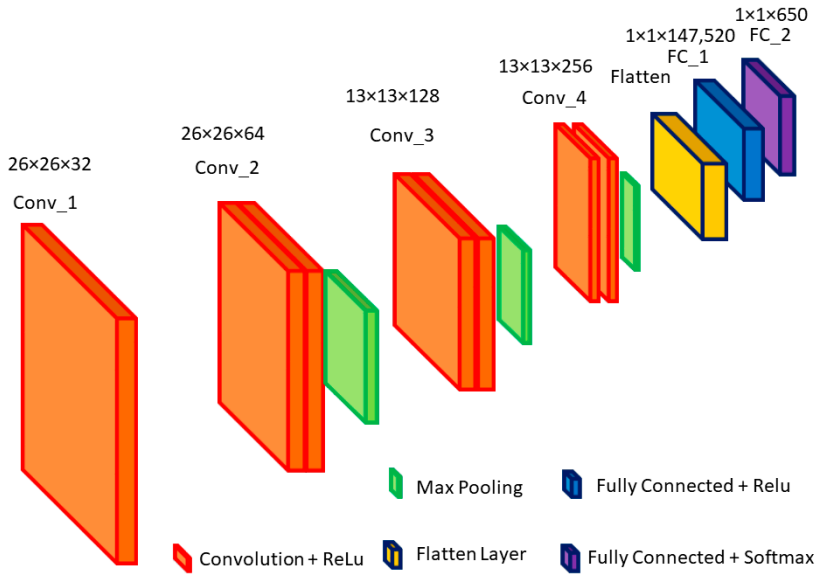

Figure 6. Proposed CNN model for character recognition.

Table 1. Details of the proposed model.

Layer (Type)	Output Shape	Param #
conv_1 (Conv 2D)	(None, 26, 26, 32)	320
conv_2 (Conv 2D)	(None, 26, 26, 64)	18,496
max_pooling2D_18 (MaxPooling2D)	(None, 13, 13, 64)	0
conv_3 (Conv 2D)	(None, 13, 13, 128)	73,856
max_pooling2D_19 (MaxPooling2D)	(None, 6, 6, 128)	0
conv_4 (Conv 2D)	(None, 6, 6, 256)	295,168
max_pooling2D_20 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
FC_1 (Dense)	(None, 64)	147,520
FC_2 (Dense)	(None, 10)	650
Total Params #		536,010
Trainable Params #		536,010
Non-Trainable Params #		0

For generalization, the same proposed model is used to classify both the English alphabets and digits. The only difference is the number of output classes defined in the last fully connected layer, which is the 'fully connected + Softmax' layer, as depicted by Figure 6, and the FC_2 layer, as presented by Table 1. The number of classes is 10 for digit recognition as depicted by the table, and the number of classes is 26 for alphabet recognition. Moreover, for extensive comparative analysis, we also analyzed how the proposed model performs with different optimizers, 'ADAM' and 'RMSprop', which also include the variation of the learning rates (LRs). This analysis helps in understanding how the model performance might vary with the change of optimizers and variation of learning rates which are discussed in detail in Section 5—Results and Analysis.

In order to avoid the difficulties posed by the problem of latency in data processing, this project utilizes Colab-pro by Google, which has a 2.20 GHz Intel Xeon Processor, 128 GB RAM, and Tesla P100 16 GB GPU. The model was designed and tested in Colab-pro, keeping in mind the factor of easy reproducibility by the research community, as Colab-pro has built-in support for GPU-enabled TensorFlow and the necessary support for CUDA acceleration.

5. Results and Analysis

We used two datasets for the handwritten recognition process: the Kaggle dataset for our English letter (A–Z) and MNIST for our numeric characters (0–9). Two optimizers were used for each of the datasets, 'ADAM' and 'RMSprop', as well as three different learning rates (LRs) of 0.001, 0.0001, and 0.00001 for each of the optimizers. This gives us six CNN models for each of the datasets and twelve models overall. To avoid confusion and repetition, we named our models. The models were named as follows for the Kaggle dataset: with a learning rate of 0.001, the model under the 'ADAM' optimizer is K1 and the one under the 'RMSprop' is K2; with a learning rate of 0.0001, the model under the 'ADAM' optimizer is K3 and the one under the 'RMSprop' is K4; with a learning rate of 0.00001, the model under the 'ADAM' optimizer is K5 and the one under the 'RMSprop' is K6. The models were named similarly for the MNIST dataset from model M1 to model M6. Our results indicated that we obtained the best result under the 'ADAM' optimizer with a learning rate of 0.00001 under the Kaggle dataset (model K5), and under 'RMSprop' with a learning rate of 0.001 for the MNIST dataset (model M2). We then calculated the *F*1 score (micro, macro, and weighted average) and obtained confusion matrices and two classification reports for the two models that give us the best accuracy for the each datasets. Figure 7 simplifies the selection of the best models for each dataset.

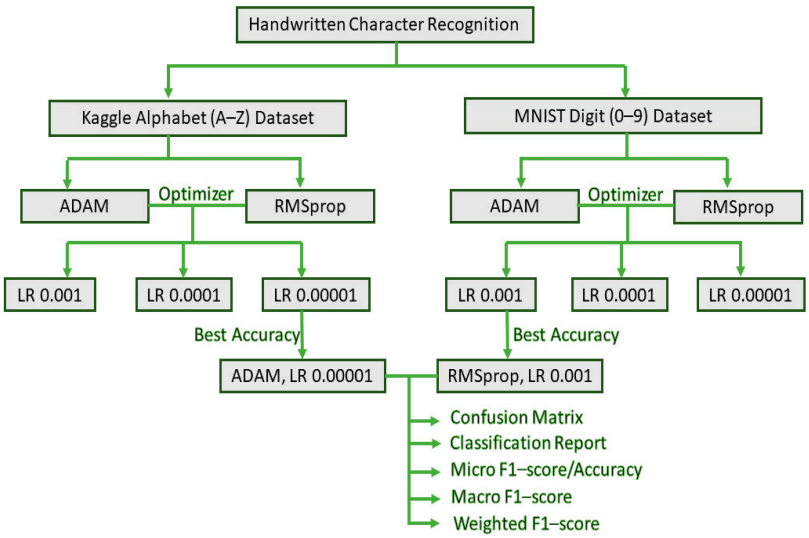

Figure 7. Best model selection from the Kaggle and MNIST datasets.

For the alphabet dataset, the overall accuracies using the 'ADAM' optimizer in the proposed CNN model for handwritten English alphabet recognition were 99.516%, 99.511%, and 99.563% for LR 0.001, LR 0.0001, and LR 0.00001, respectively. The same model using 'RMSprop' achieved the accuracy of 99.292%, 99.108%, and 99.191%, respectively, by LR 0.001, LR 0.0001, and LR 0.00001. These results clearly show that, in terms of accuracy, the model using the 'ADAM' optimizer with LR 0.00001, named as model K5, performs better than the other proposed models. It is clear that all the proposed six models for character recognition achieved above 99.00% overall accuracy.

For the digit dataset, the overall accuracies using 'RMSprop' for handwritten digit recognition were 99.642%, 99.452%, and 98.142% for LR 0.001, LR 0.0001, and LR 0.00001, respectively. The same model using the 'ADAM' optimizer achieved accuracies of 99.571%, 99.309%, and 98.142% for LR 0.001, LR 0.0001, and LR 0.00001, respectively. Figures 8 and 9 depict validation accuracies and Figures 10 and 11 show the validation losses of all the twelve models with the Kaggle and MNIST dataset, respectively. It is clear that overall accuracy decreases with the decrease in learning rate (LR). This confirms that the model using 'RMSprop' with LR 0.001, named as model M2, outperformed the other proposed models in terms of accuracy. From Figures 9 and 11, it can be clearly observed that no overfitting happens for the digit recognition or for alphabet recognition; overfitting occurs when 'RMSprop' is used, which is depicted in Figures 8d–f and 10d–f. Overfitting occurs when the model performs fine on the training data but does not perform exactly in the testing set. Here, the model learns the unnecessary information within the dataset as it trains for a long time on the training data.

The performance evaluation of the models is more obvious and explicit from the matrices of specificity, recall, precision, *F*1 score, and support. The possible outcomes obtained by the confusion matrix (CM) calculate the performance of these matrices. This CM has four different outcomes: total false positive (TFP), total false negative (TFN), total true positive (TTP), and total true negative (TTN). The CM sets up nicely to compute the per-class values of recall, precision, specificity, and *F*1 score for each of the datasets.

Let us consider the scenario where we want the model to detect the letter 'A'. For simplification, let us also assume that each of the 26 letters in the alphabet (A–Z) has 100 images for each of the letters, totaling 2600 images altogether. If we assume that the model accurately identifies the images of the letter 'A' in 97 out of 100 images, then we say that the accuracy of the model is 97%. Thus, we can also conclude that the total number of true positives (TTPs) is 97. Under the same assumptions as above, if the letter 'O' is incorrectly identified as 'A', then this would tell us that the number of total false positives (TFPs) in this case would be 1. If the letter 'A' has been misidentified as 'O' three times in the model, then the total number of false negatives (TFNs) for this model is 3. The rest of the 2499 images of the 2600 images are then considered as the total true negative (TTN). Figures 12 and 13 show the confusion matrices for the best two models (model K5 for letter recognition and model M2 for digit recognition) established in terms of overall performance that were trained and validated with the Kaggle and MNIST datasets, respectively.

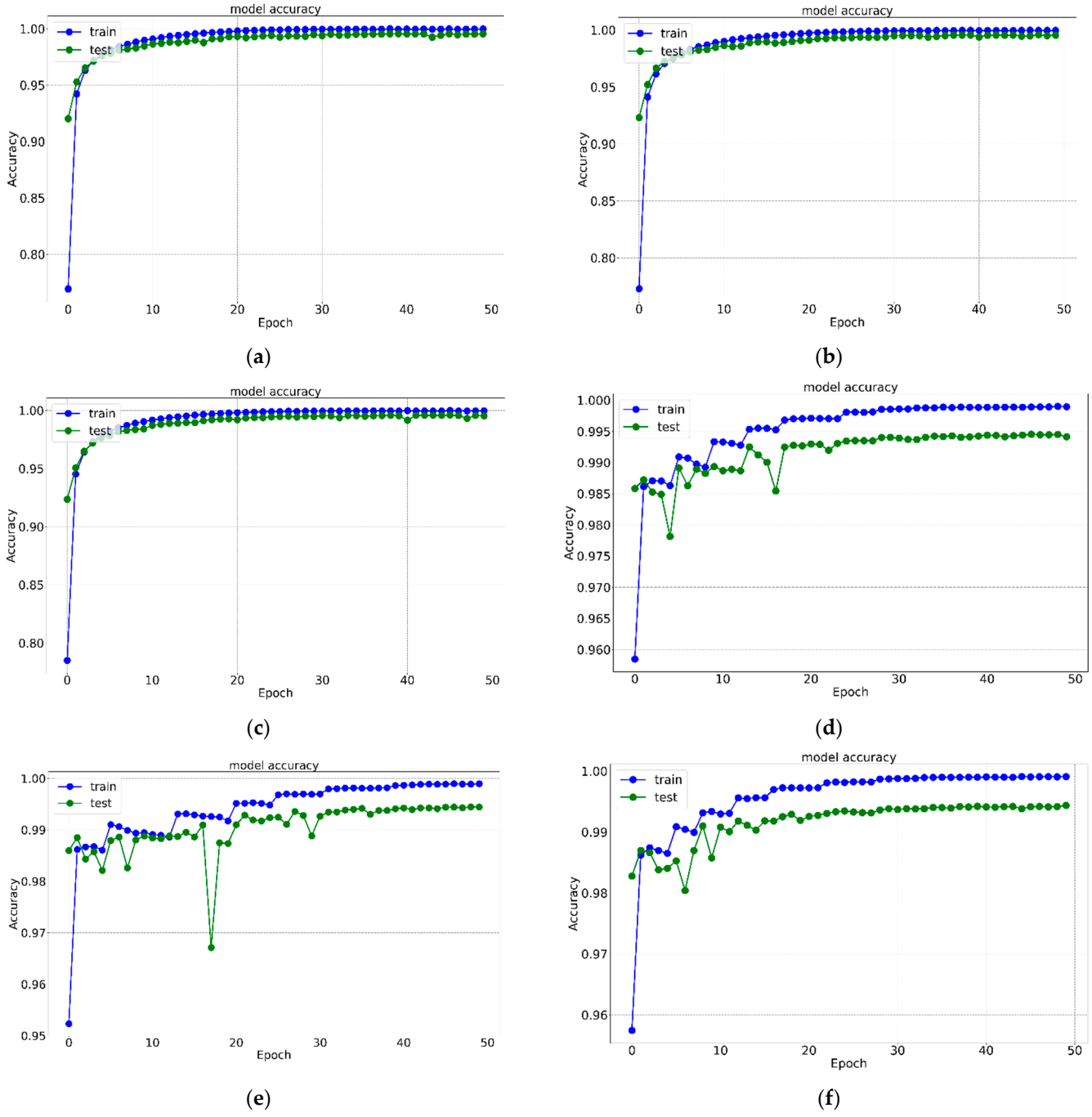

Figure 8. Validation accuracy of the six models for English alphabet recognition. (**a**) Optimizer—'ADAM'; learning rate—0.001. (**b**) Optimizer—'ADAM'; learning rate—0.0001. (**c**) Optimizer—'ADAM'; learning rate—0.00001. (**d**) Optimizer—'RMSprop'; learning rate—0.001. (**e**) Optimizer—'RMSprop'; learning rate—0.0001. (**f**) Optimizer—'RMSprop'; learning rate—0.00001.

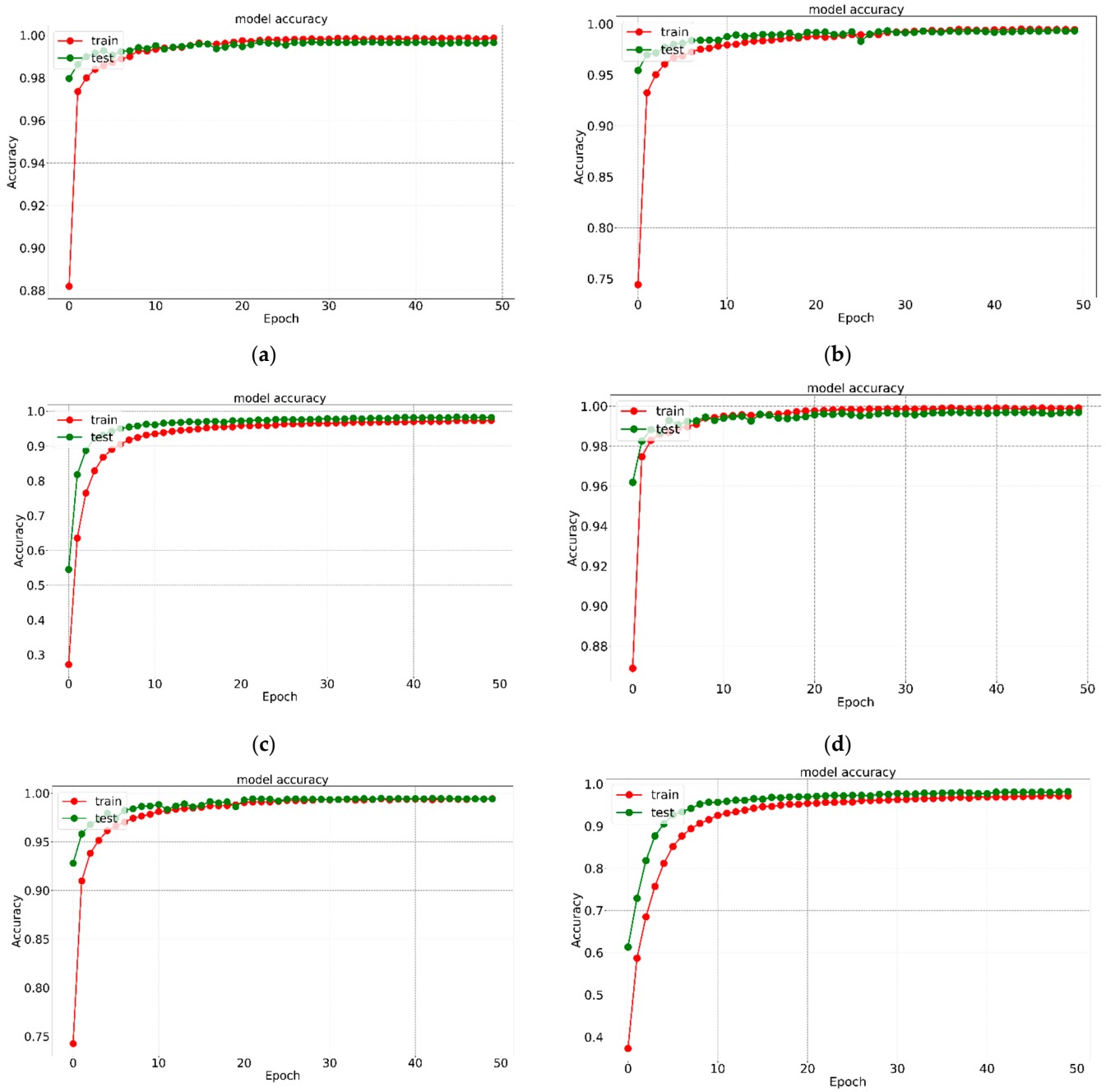

Figure 9. Validation accuracy of the six models for digit (0–9) recognition. (**a**) Optimizer—'ADAM'; learning rate—0.001. (**b**) Optimizer—'ADAM'; learning rate—0.0001. (**c**) Optimizer—'ADAM'; learning rate—0.00001. (**d**) Optimizer—'RMSprop'; learning rate—0.001. (**e**) Optimizer—'RMSprop'; learning rate—0.0001. (**f**) Optimizer—'RMSprop'; learning rate—0.00001.

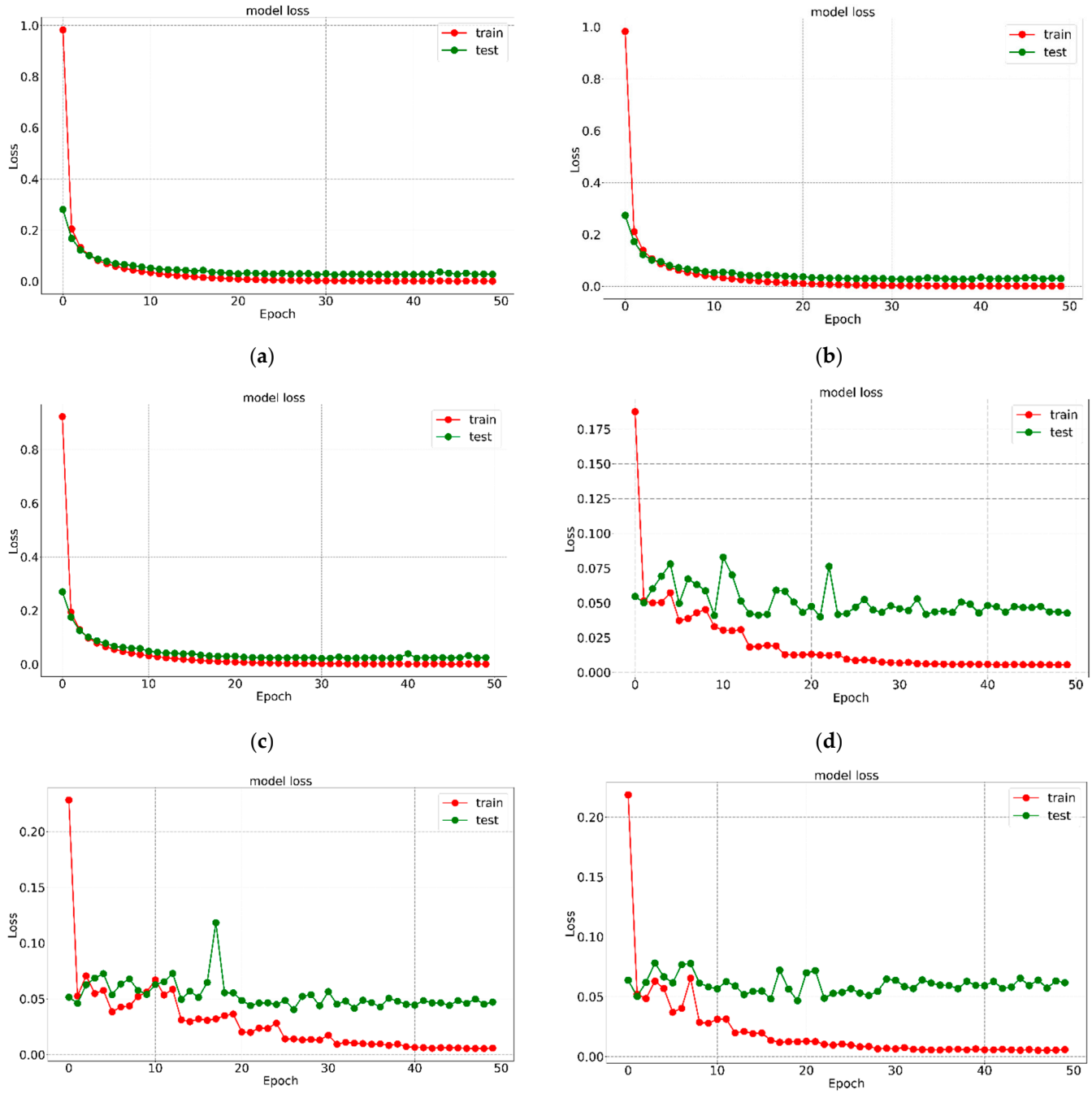

Figure 10. Validation loss of the six models for English alphabet recognition. (**a**) Optimizer—'ADAM'; learning rate—0.001. (**b**) Optimizer—'ADAM'; learning rate—0.0001. (**c**) Optimizer—'ADAM'; learning rate—0.00001. (**d**) Optimizer—'RMSprop'; learning rate—0.001. (**e**) Optimizer—'RMSprop'; learning rate—0.0001. (**f**) Optimizer—'RMSprop'; learning rate—0.00001.

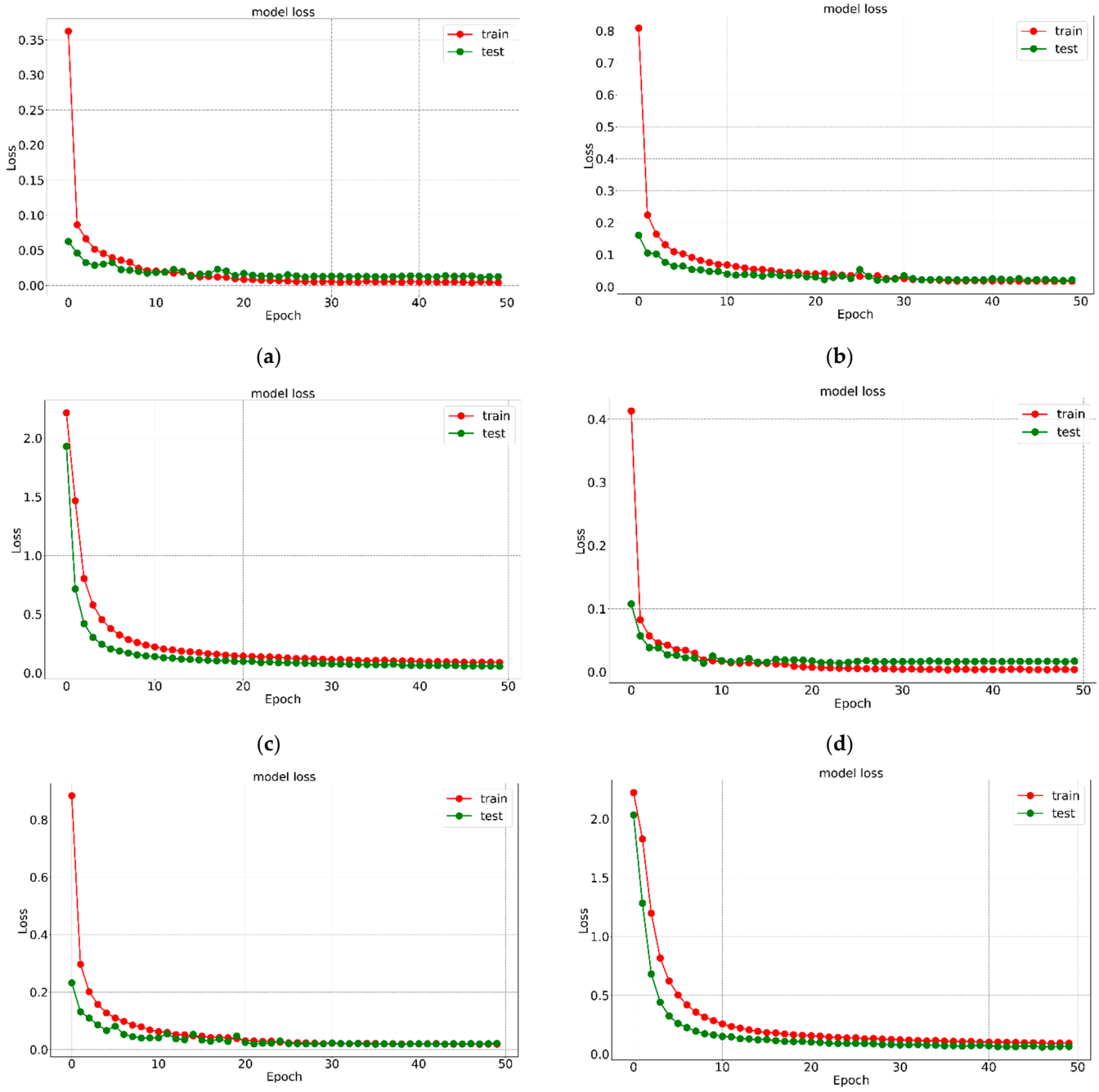

Figure 11. Validation loss of the proposed six models for digit (0–9) recognition. (**a**) Optimizer—'ADAM'; learning rate—0.001. (**b**) Optimizer—'ADAM'; learning rate—0.0001. (**c**) Optimizer—'ADAM'; learning rate—0.00001. (**d**) Optimizer—'RMSprop'; learning rate—0.001. (**e**) Optimizer—'RMSprop'; learning rate—0.0001. (**f**) Optimizer—'RMSprop'; learning rate—0.00001.

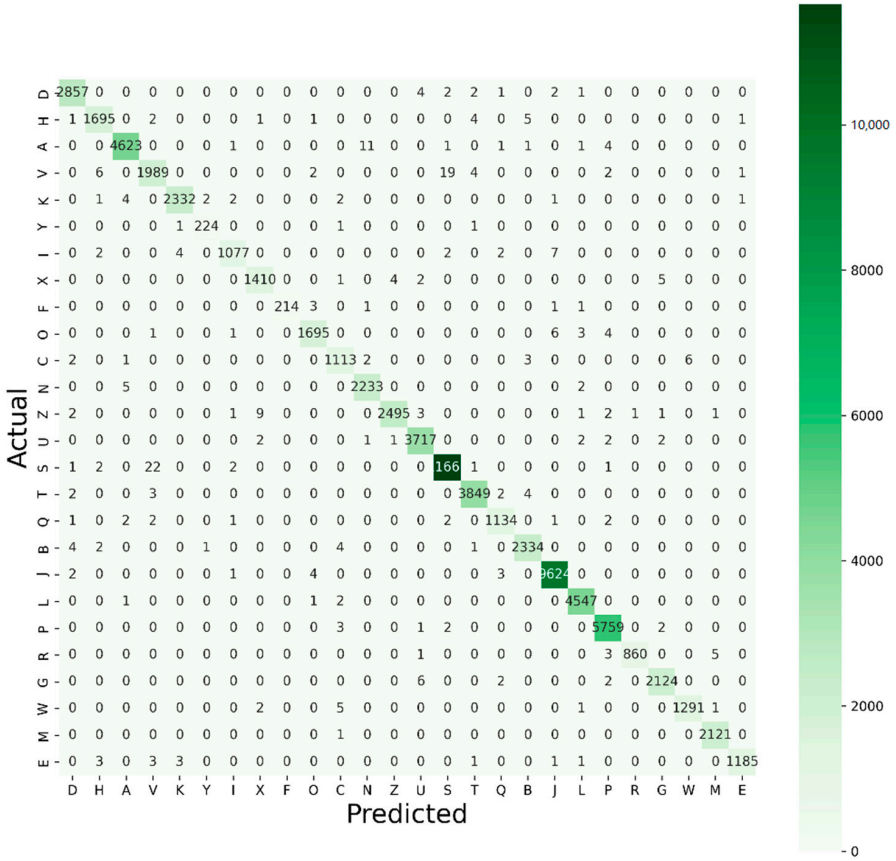

Figure 12. Confusion matrix of model K5 for A–Z recognition.

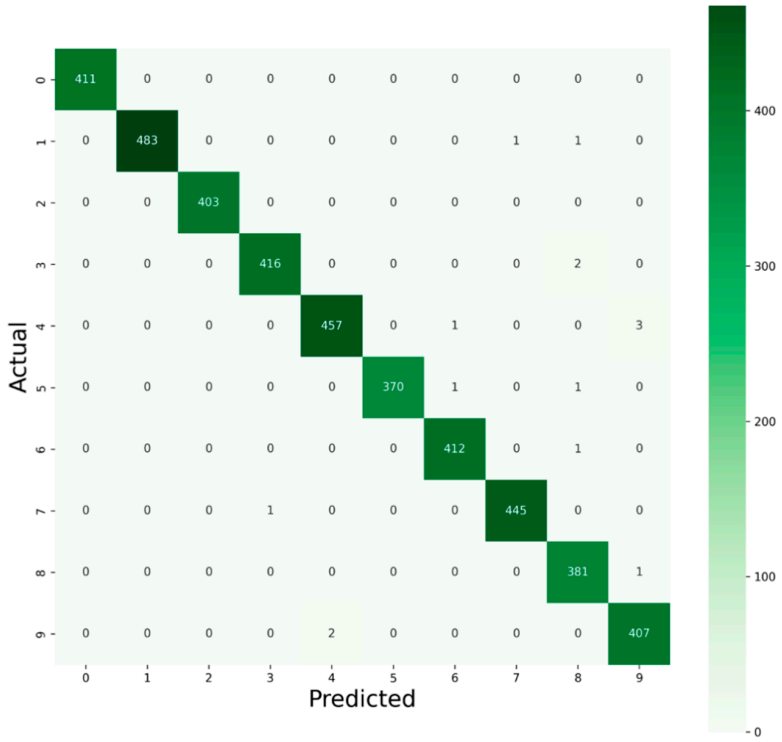

Figure 13. Confusion matrix of model M2 for 0–9 recognition.

Precision deals with the percentage of the relevant results, whereas accuracy states how close the real values are to the generated values. Sensitivity, identified as recall and true negative rate, known as specificity, are other important factors for investigating a CNN model. The *F*1 score is the weighted average of the combination of both precision and recall. Equations (1)–(5) represent accuracy, specificity, recall, precision, and *F*1 score, respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$Specificity = \frac{TN}{TN + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$F1\ Score = 2 * \left(\frac{Precision * Recall}{Precision + Recall} \right) \tag{5}$$

With the Kaggle dataset of 74,490 testing images for letter recognition, using 'ADAM' optimizer model K1 detects 74,130 TTP and 360 TFP images, model K3 finds 74,126 TTP and 364 TFP images, whereas model K5 detects 74,165 TTP and 325 TFP images. Then, again, the models using 'RMSprop' underperform in identifying the TFP cases, which is above 500 for each model, while the TTP cases detected using 'RMSprop' are 73,963, 73,826, and 73,888 by models K2, K4, and K6, respectively. The recall of model K5 is 99.56%, performing better than the other investigated models. In contrast, model K4 attained the lowest recall percentage of 99.1% (which is also above 99%) in comparison with others. From the confusion matrices, it was noted that all the proposed models achieved the specificity of 99% and model K5 performed supremely for recognizing handwritten letters.

It is important to recall that, in multiclass classification, we compute the *F*1 score for each class in a one-versus-the-rest (OvR) tactic, instead of a single overall *F*1 score, as perceived in binary classification. Therefore, the total true negative (TTN) number is vital to evaluate the proposed model. With the Kaggle dataset, model K5 detected 1,861,925 TTN images, which was the highest number detected; contrarily, model K4 detected the lowest number, at 1,861,586 TTN images. The overall performance and the total number of TTN images detected of the proposed model K5 showed better results than the other models, as it is always expected that higher TTN cases will be obtained for each class while recognizing specific letters. The accuracy of the proposed model K5 was 99.563%, with precision and recall values of 99.5% for both the parameters, which is the highest handwritten alphabet recognition accuracy known to the authors for the Kaggle dataset.

Now, for the MNIST dataset for digit recognition, model M1, model M2, and model M3 performed with accuracies of 99.571%, 99.309%, and 98.238%, respectively, whereas model M2, model M4, and model M6 performed with accuracies of 99.642%, 99.452%, and 98.142%, respectively. Model M2 ('RMSprop' with LR 0.001) achieved the highest precision of 99.6, whereas model M6 obtained the lowest precision of 98.1 (which is also above 98 %). Similarly, the highest value of recall was 0.9964 by model M2, and the lowest recall value of 0.9814 was by model M6. It was also seen that, from confusion matrices, with MNIST 5000 test patterns for digit recognition, model M2 had the highest, preeminent performance in comparison with other models, because it found 4185 TTP and 15 TFP images, and had the highest TTN case of 37,785. Model M6 performed poorest when compared with other models; the TTP and TFP cases are 4122 and 78, respectively, and the TTN cases decreased by 65 images compared with model M2; however, while recognizing specific digits, it was expected to always obtain higher TTN cases for each class. Table 2 shows the comparative results with the Kaggle and MNIST datasets of all the models.

Table 2. Confusion matrix parameters of all the investigated models.

Data-Set	Learning Rate	Optimizer	Model Name	Precision (%)	Specificity (%)	Recall (%)	TFP	TFN	TTP	TTN	Overall Accuracy (%)
Kaggle Alphabet Recognition	0.001	'ADAM'	K1	99.4	99.98	99.51	360	360	74,130	1,861,890	99.516
		RMS_prop	K2	99.2	99.97	99.29	527	527	73,963	1,861,723	99.292
	0.0001	'ADAM'	K3	99.5	99.98	99.51	364	364	74,126	1,861,886	99.511
		RMS_prop	K4	99.0	99.96	99.10	664	664	73,826	1,861,586	99.108
	0.00001	'ADAM'	K5	99.5	99.98	99.56	325	325	74,165	1,861,925	99.563
		RMS_prop	K6	99.1	99.96	99.19	602	602	73,888	1,861,648	99.191
MNIST Digit Recognition	0.001	'ADAM'	M1	99.5	99.95	99.57	22	22	4178	37,778	99.571
		RMS_prop	M2	99.6	99.96	99.64	15	15	4185	37,785	99.642
	0.0001	'ADAM'	M3	99.2	99.92	99.30	29	29	4171	37,771	99.309
		RMS_prop	M4	99.4	99.93	99.45	23	23	4177	37,777	99.452
	0.00001	'ADAM'	M5	98.2	99.80	98.23	74	74	4126	37,726	98.238
		RMS_prop	M6	98.1	99.79	98.14	78	78	4122	37,722	98.142

To classify both the English alphabets and digits, using two different optimizers, four 2D convolutional layers are kept the same and unchanged in the proposed CNN model. The only difference is the number of output classes, defined in the last fully connected layer, which is the 'fully connected + Softmax' layer. The number of classes is 10 and 26 for digit and alphabet recognition, respectively. It is clearly seen from Table 2 that the 'ADAM' optimizer performs better than 'RMSprop' for letter recognition, whereas, for digit recognition, 'RMSprop' is more suitable. These optimizers are used here to obtain fast-tracked results by changing the attributes of the proposed neural networks. For alphabet recognition, with the Kaggle dataset, it shows that the models (i.e., K1–K6) perform better with the decrement of one of the hyper-parameters, the learning rate (LR); contrariwise, with the MNIST dataset, for digit recognition, it displays that the models (model M1–model M6) perform well with the increment of the learning rate, e.g., the overall accuracy increases to 1.53% while using 'RMSprop', and the learning rate increases from 0.00001 to 0.001 for the MNIST dataset.

It can be seen from Figures 12 and 13 that the distribution of the dataset is imbalanced. Hence, only the accuracy would be ineffective in judging model performance and so the classification report (CR) is indispensable for an analytical understanding of the model predictions. The advanced performances can be analyzed to a great extent with a classification report (CR) for the best two proposed models (model M2 and model K5), which are presented in Tables 3 and 4, respectively.

Table 3. Classification report of model M2 for 0–9 recognition.

Digit (0–9)	Precision /Class	Recall /Class	*F*1 Score /Class	Support /Class	Support Proportion/Class
class 0	1.00	1.00	1.00	411	0.098
class 1	1.00	1.00	1.00	485	0.115
class 2	1.00	1.00	1.00	403	0.096
class 3	1.00	1.00	1.00	418	0.1
class 4	1.00	0.99	0.99	461	0.11
class 5	1.00	0.99	1.00	372	0.089
class 6	1.00	1.00	1.00	413	0.098
class 7	1.00	1.00	1.00	446	0.106
class 8	0.99	1.00	0.99	382	0.091
class 9	0.99	1.00	0.99	409	0.097
Total				4200	1.00

Table 4. Classification report of model K5 for A–Z recognition.

Letters (A–Z)	Precision /Class	Recall /Class	*F*1 score /Class	Support /Class	Support Proportion /Class
class A	0.99	0.99	0.99	1459	0.02
class B	1.00	0.99	1.00	4747	0.064
class C	0.99	1.00	0.99	2310	0.031
class D	1.00	1.00	1.00	5963	0.08
class E	0.99	0.99	0.99	1986	0.027
class F	0.99	0.99	0.99	1161	0.016
class G	1.00	0.99	0.99	1712	0.023
class H	0.99	1.00	0.99	2291	0.031
class I	1.00	1.00	1.00	3894	0.052
class J	0.99	1.00	0.99	2724	0.037
class K	0.99	0.99	0.99	2315	0.031
class L	0.98	0.99	0.99	1109	0.015
class M	1.00	0.99	1.00	3841	0.052
class N	1.00	1.00	1.00	11,524	0.155
class O	0.99	0.99	0.99	2488	0.033
class P	0.99	0.99	0.99	1235	0.017
class Q	1.00	1.00	1.00	4518	0.061
class R	0.99	0.99	0.99	1226	0.016
class S	1.00	0.98	0.99	229	0.003
class T	1.00	0.99	0.99	870	0.012
class U	1.00	0.99	1.00	2045	0.027
class V	1.00	1.00	1.00	9529	0.128
class W	0.99	0.98	0.99	1145	0.015
class X	0.99	0.99	0.99	2165	0.029
class Y	0.97	0.96	0.97	249	0.003
class Z	0.99	0.99	0.99	1755	0.024
Total				74,490	1.00

The columns marked in yellow in Tables 3 and 4 indicate the score for each of the 10 classes for digits (0–9) and each of the 26 classes for alphabet, A–Z. In our case, using multiclass classification calculation, we pursued averaging methods for the *F*1 score, resulting in different average scores, i.e., micro, macro, and weighted averaging. Macro averaging is possibly the most straightforward among the averaging methods. Regardless of their support values, this method treats all classes without differentiation. Support denotes the number of actual occurrences of the class in the dataset. The macro *F*1 score is totaled by taking the unweighted arithmetic mean of all the per-class *F*1 scores. We calculated macro *F*1 scores of 0.998 and 0.992 for model *M*2 and model *K*5, respectively.

$$Macro\ F1\ score\ (model\ M2) = \frac{8*1+2*99}{10} = 0.998$$

$$Macro\ F1\ score\ (model\ K5) = \frac{8*1+17*99+.97}{26} = 0.992$$

The weighted average *F*1 score was computed by counting the mean of all per-class *F*1 scores while considering each classes' support. This average refers to the proportion of each classes' support, relative to the sum of all support values. In our case of multiclass classification, using Equation (6), the calculated values of weighted *F*1 score are 0.997 and 0.996 for model M2 and model K5, respectively. Micro averaging computes a universal

average *F*1 score by taking the sums of the TTP, TFN, and TFP. Micro *F*1 score is computed using Equation (7), which is derived from Equation (5). Table 5 shows the micro, macro, and weighted average of *F*1 score for the best two proposed models (model M2 and model K5). The performance of a classification model is evaluated by the following well-accepted F-measurement matrix:

$$Weighted\ F1\ score = \sum_{a}^{b}(Per\ class\ F1\ score * Support\ Proportion) \quad (6)$$

$$Micro\ F1\ score = \frac{TP}{TP + \frac{1}{2}\ (FP + FN)} \quad (7)$$

Table 5. Micro, macro, and weighted average of *F*1 score for the best two proposed models.

F-Measure	Model M2 (Digit Recognition)	Model K5 (Letter Recognition)
Micro *F*1 score	0.996	0.995
Macro *F*1 score	0.998	0.992
Weighted *F*1 score	0.997	0.996

Micro *F*1 score works well with the balanced dataset. In this project, as the datasets are imbalanced, the macro average would be an ideal choice where all classes are equally significant, because it treats all classes equally. The weighted average is preferred if we want to assign greater contribution to classes with more examples, because each classes' contribution to the *F*1 average is weighted by its size.

We obtained good results with multiclass classification. The proposed CNN model (with four convolutional layers, three max-pooling layers, and two dense layers) for handwritten recognition was approached by keeping in mind that it should not start overfitting. However, this work shows how different learning rates and optimizers play a part in the models' performances. Additionally, classification reports are presented for micro, macro, and weighted averages. A comparative analysis of how the best two proposed models perform with other distinguished models in recent years by different researchers is shown in Table 6.

Table 6. Comparison of different learning models' approaches based on dataset, preprocessing, and accuracy.

Sl. No.	Author(s)	Publication Year	Approach	Dataset	Preprocessing	Results
1.	Mor et al. [63]	2019	Two convolutional layers and one dense layer.	EMNIST	X	87.1%
2.	Alom et al. [64]	2017	CNN with dropout and Gabor Filters.	CMATERdb 3.1.1	Raw images passed to Normalization	98.78%
3.	Sabour et al. [65]	2019	A CNN with 3 convolutional layers and two capsule layers.	EMNIST	X	95.36% (Letters) 99.79% (Digits)
4.	Dos Santos et al. [66]	2019	Deep convolutional extreme learning machine.	EMNIST Digits	X	99.775%.
5.	Adnan et al. [67]	2018	Deep Belief Network (DBN), Stacked Auto encoder (AE), DenseNet	CMATERdb 3.11	600 images are rescaled to 32 × 32 pixels.	99.13% (Digits) 98.31% (Alphabets) 98.18% (Special Character)
6.	W. Xue et al. [68]	2020	Three CNN were combined into a single feature map for classification.	UC Merced, AID, and NWPU-RESISC45	X	AID: 93.47% UC Merced: 98.85%, NWPU-RESISC45: 95%
7.	D.S.Prashanth et al. [69]	2020	1. CNN, 2. Modified Lenet CNN (MLCNN) and 3. Alexnet CNN (ACNN).	Own dataset of 38,750 images	X	CNN: 94% MLCNN: 99% ACNN: 98%
8.	D.S.Joshi and Risodkar [70]	2018	K-NN classifier and Neural Network	Own dataset with 30 samples	RGB to gray conversion, skew correction, filtering, morphological operation	78.6%
9.	Ptucha et al. [51]	2020	Introduced an intelligent character recognition (ICR) system	IAM RIMES lexicon	X	99%
10.	Shibaprasad et al. [71]	2018	Convolutional Neural Network (CNN) architecture	1000-character samples	Resized all images to 28 × 28 pixels.	99.40%
11.	Yu Weng and Cnulei Xia [72]	2019	Deep Neural Network (DNNs)	Own dataset of 400 types of pictures	Normalized to 52 × 52 pixels.	93.3%
12.	Gan et al. [73]	2019	1-D CNN	ICDAR-2013 IAHCC-UC AS2016	Chinese character images rescaled into 60 × 60-pixel size.	98.11% (ICDAR-2013) 97.14% (IAHCC-UCA2016)
13.	Kavitha et al. [45]	2019	CNN (5 convolution layers, 2 max pooling layers, and fully connected layers)	HPL-Tamil-is o-char	RGB to gray conversion	97.7%.
14.	Saha et al. [74]	2019	Divide and Merge Mapping (DAMM)	Own dataset with 1,66,105 images	Resize all images to 128 × 128.	99.13%
15.	Y. B. Hamdan et al. [75]	2021	Support vector machine (SVM) classifiers network graphical methods.	MNIST, CENPARMI	X	94%

Table 6. *Cont.*

Sl. No.	Author(s)	Publication Year	Approach	Dataset	Preprocessing	Results
16.	Ukil et al. [76]	2019	CNNs	PHD Indic_11	RGB to grayscale conversion and resized image to 28 × 28 pixels.	95.45%
17.	Cavalin et al. al. [77]	2019	A hierarchical classifier by the confusion matrix of flat classifier	EMNIST	X	99.46% (Digits) 93.63% (Letters)
18.	Tapotosh Ghosh et al. [58]	2021	InceptionResNetV2, DenseNet121, and InceptionNetV3	CMATERdb	The images were first converted to B&W 28 × 28 form with white as the foreground color.	97.69%
19.	Proposed Model	2022	CNN using 'RMSprop' and 'ADAM' optimizer with four convolutional layers, three max pooling and two dense layers are used for three different Learning rates (LR 0.001, LR 0.0001 and LR 0.00001) for multiclass classification.	MNIST: 60,000 training, 10,000 testing images. Kaggle: 297,000 training, 74,490 testing images.	Each digit/letter is of a uniform size and by computing the center of mass of the pixels, each binary image of a handwritten digit is centered into a 28 × 28 image.	99.64% (Digits) Macro *F*1 score average: 0.998 99.56% (Letters) Macro *F*1 score average: 0.992

6. Conclusions

In modern days, applications of handwritten character recognition (HRC) systems are flourishing. In this paper, to address HCR systems with multiclass classification, a CNN-based model is proposed that achieved exceptionally good results with this multiclass classification. The CNN models were trained with the MNIST digit dataset, which is shaped with 60,000 training and 10,000 testing images. They were also trained with the substantially larger Kaggle alphabet dataset, which comprises over 297,000 training images and a test set which is shaped on testing over 74,490 images. For the Kaggle dataset, the overall accuracies using the 'ADAM' optimizer were 99.516%, 99.511%, and 99.563% for learning rate (LR) 0.001, LR 0.0001, and LR 0.00001, respectively. Meanwhile, the same model using 'RMSprop' achieved accuracies of 99.292%, 99.108%, and 99.191%, respectively, by LR 0.001, LR 0.0001, and LR 0.00001. For the MNIST dataset, the overall accuracies using 'RMSprop' were 99.642%, 99.452%, and 98.142% for LR 0.001, LR 0.0001, and LR 0.00001, respectively. Meanwhile, the same model using the 'ADAM' optimizer achieved accuracies of 99.571%, 99.309%, and 98.142% with LR 0.001, LR 0.0001, and LR 0.00001, respectively. It can be easily understood that, for alphabet recognition, accuracy decreases with the increase in learning rate (LR); contrarily, overall accuracy is proportionately related to LR for digit recognition. In addition, precision, recall, specificity, and $F1$ score were measured from confusion matrices. Of all the discussed twelve models, the model using the 'ADAM' optimizer with LR 0.00001 obtained a recall value of 99.56%, and the model with LR 0.001 with the 'RMSprop' optimizer obtained the recall value of 99.64%; therefore, these two models excel other models for the Kaggle and MNIST datasets, respectively. As the distribution of the datasets is imbalanced, only the accuracy would be ineffective in evaluating the models; therefore, classification reports (CR) indicating the $F1$ score for every 10 classes for digits (0–9) and every 26 classes for alphabet (A–Z) were included for the predictions of the best two proposed models. From the CR, we achieved micro, macro, and weighted $F1$ scores of 0.996 and 0.995, 0.998 and 0.992, and 0.997 and 0.996 for the MNIST and Kaggle datasets, respectively. Furthermore, the obtained results of best two models presented here were compared with the results of other noticeable works in this arena. Considering future work, we intend to include several feature extraction methods by applying a similar framework to that proposed here to more complex languages, such as Korean, Chinese, Finnish, and Japanese.

Author Contributions: Conceptualization, N.S., K.F.H. and A.A.; methodology, N.S. and K.F.H.; software, N.S. and K.F.H.; validation, N.S. and A.A.; formal analysis, N.S.; investigation, N.S. and K.F.H.; writing—original draft preparation, N.S. and K.F.H.; writing—review and editing, N.S., K.F.H., V.P.Y. and A.A.; visualization, N.S. and V.P.Y.; supervision, V.P.Y. and A.A.; project administration, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Priya, A.; Mishra, S.; Raj, S.; Mandal, S.; Datta, S. Online and offline character recognition: A survey. In Proceedings of the International Conference on Communication and Signal Processing, (ICCSP), Melmaruvathur, Tamilnadu, India, 6–8 April 2016; pp. 967–970.
2. Gunawan, T.S.; Noor, A.F.R.M.; Kartiwi, M. Development of english handwritten recognition using deep neural network. *Indones. J. Electr. Eng. Comput. Sci.* **2018**, *10*, 562–568. [CrossRef]
3. Vinh, T.Q.; Duy, L.H.; Nhan, N.T. Vietnamese handwritten character recognition using convolutional neural network. *IAES Int. J. Artif. Intell.* **2020**, *9*, 276–283. [CrossRef]

4. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893.
5. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
6. Xiao, J.; Zhu, X.; Huang, C.; Yang, X.; Wen, F.; Zhong, M. A New Approach for Stock Price Analysis and Prediction Based on SSA and SVM. *Int. J. Inf. Technol. Decis. Mak.* **2019**, *18*, 35–63. [CrossRef]
7. Wang, D.; Huang, L.; Tang, L. Dissipativity and synchronization of generalized BAM neural networks with multivariate discontinuous activations. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 3815–3827. [CrossRef]
8. Kuang, F.; Zhang, S.; Jin, Z.; Xu, W. A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection. *Soft Comput.* **2015**, *19*, 1187–1199. [CrossRef]
9. Choudhary, A.; Ahlawat, S.; Rishi, R. A binarization feature extraction approach to OCR: MLP vs. RBF. In Proceedings of the International Conference on Distributed Computing and Technology (ICDCIT), Bhubaneswar, India, 6–9 February 2014; Springer: Cham, Switzerland, 2014; pp. 341–346.
10. Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* **1980**, *36*, 193–202. [CrossRef]
11. Ahlawat, S.; Choudhary, A.; Nayyar, A.; Singh, S.; Yoon, B. Improved handwritten digit recognition using convolutional neural networks (Cnn). *Sensors* **2020**, *20*, 3344. [CrossRef]
12. Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.; LeCun, Y. What is the best multi-stage architecture for object recognition? In Proceedings of the IEEE 12th International Conference on Computer Vision (ICCV), Kyoto, Japan, 29 September–2 October 2009; pp. 2146–2153.
13. Cireşan, D.C.; Meier, U.; Masci, J.; Gambardella, L.M.; Schmidhuber, J. High-Performance Neural Networks for Visual Object Classification. *arXiv* **2011**, arXiv:1102.0183v1.
14. Ciresan, D.; Meier, U.; Schmidhuber, J. Multi-column deep neural networks for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 3642–3649.
15. Niu, X.X.; Suen, C.Y. A novel hybrid CNN-SVM classifier for recognizing handwritten digits. *Pattern Recognit.* **2012**, *45*, 1318–1325. [CrossRef]
16. Qu, X.; Wang, W.; Lu, K.; Zhou, J. Data augmentation and directional feature maps extraction for in-air handwritten Chinese character recognition based on convolutional neural network. *Pattern Recognit. Lett.* **2018**, *111*, 9–15. [CrossRef]
17. Alvear-Sandoval, R.F.; Figueiras-Vidal, A.R. On building ensembles of stacked denoising auto-encoding classifiers and their further improvement. *Inf. Fusion* **2018**, *39*, 41–52. [CrossRef]
18. Demir, C.; Alpaydin, E. Cost-conscious classifier ensembles. *Pattern Recognit. Lett.* **2005**, *26*, 2206–2214. [CrossRef]
19. Choudhary, A.; Ahlawat, S.; Rishi, R. A Neural Approach to Cursive Handwritten Character Recognition Using Features Extracted from Binarization Technique. *Stud. Fuzziness Soft Comput.* **2015**, *319*, 745–771. [CrossRef]
20. Cai, Z.W.; Huang, L.H. Finite-time synchronization by switching state-feedback control for discontinuous Cohen–Grossberg neural networks with mixed delays. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 1683–1695. [CrossRef]
21. Zeng, D.; Dai, Y.; Li, F.; Sherratt, R.S.; Wang, J. Adversarial learning for distant supervised relation extraction. *Comput. Mater. Contin.* **2018**, *55*, 121–136. [CrossRef]
22. Long, M.; Zeng, Y. Detecting iris liveness with batch normalized convolutional neural network. *Comput. Mater. Contin.* **2019**, *58*, 493–504. [CrossRef]
23. Huang, C.; Liu, B. New studies on dynamic analysis of inertial neural networks involving non-reduced order method. *Neurocomputing* **2019**, *325*, 283–287. [CrossRef]
24. Xiang, L.; Li, Y.; Hao, W.; Yang, P.; Shen, X. Reversible natural language watermarking using synonym substitution and arithmetic coding. *Comput. Mater. Contin.* **2018**, *55*, 541–559. [CrossRef]
25. Huang, Y.S.; Wang, Z.Y. Decentralized adaptive fuzzy control for a class of large-scale MIMO nonlinear systems with strong interconnection and its application to automated highway systems. *Inf. Sci. (Ny).* **2014**, *274*, 210–224. [CrossRef]
26. Ahlawat, S.; Rishi, R. A Genetic Algorithm Based Feature Selection for Handwritten Digit Recognition. *Recent Pat. Comput. Sci.* **2018**, *12*, 304–316. [CrossRef]
27. Hinton, G.E.; Osindero, S.; Teh, Y.W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef] [PubMed]
28. Pham, V.; Bluche, T.; Kermorvant, C.; Louradour, J. Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), Heraklion, Greece, 1–4 September 2014; pp. 285–290.
29. Lang, G.; Li, Q.; Cai, M.; Yang, T.; Xiao, Q. Incremental approaches to knowledge reduction based on characteristic matrices. *Int. J. Mach. Learn. Cybern.* **2017**, *8*, 203–222. [CrossRef]
30. Tabik, S.; Alvear-Sandoval, R.F.; Ruiz, M.M.; Sancho-Gómez, J.L.; Figueiras-Vidal, A.R.; Herrera, F. MNIST-NET10: A heterogeneous deep networks fusion based on the degree of certainty to reach 0.1% error rate. ensembles overview and proposal. *Inf. Fusion* **2020**, *62*, 73–80. [CrossRef]
31. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef]

32. Liang, T.; Xu, X.; Xiao, P. A new image classification method based on modified condensed nearest neighbor and convolutional neural networks. *Pattern Recognit. Lett.* **2017**, *94*, 105–111. [CrossRef]
33. Sueiras, J.; Ruiz, V.; Sanchez, A.; Velez, J.F. Offline continuous handwriting recognition using sequence to sequence neural networks. *Neurocomputing* **2018**, *289*, 119–128. [CrossRef]
34. Simard, P.Y.; Steinkraus, D.; Platt, J.C. Best practices for convolutional neural networks applied to visual document analysis. In Proceedings of the International Conference on Document Analysis and Recognition(ICDAR), Edinburgh, UK, 3–6 August 2003; Volume 3, pp. 958–963.
35. Wang, T.; Wu, D.J.; Coates, A.; Ng, A.Y. End-to-end text recognition with convolutional neural networks. In Proceedings of the 21st-International Conference on Pattern Recognition, Tsukuba, Japan, 11–15 November 2012; pp. 3304–3308.
36. Shi, B.; Bai, X.; Yao, C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2298–2304. [CrossRef]
37. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [CrossRef]
38. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
39. Wu, Y.C.; Yin, F.; Liu, C.L. Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognit.* **2017**, *65*, 251–264. [CrossRef]
40. Xie, Z.; Sun, Z.; Jin, L.; Feng, Z.; Zhang, S. Fully convolutional recurrent network for handwritten Chinese text recognition. In Proceedings of the International Conference on Pattern Recognition, Cancun, Mexico, 4–8 December 2016; pp. 4011–4016.
41. Liu, C.L.; Yin, F.; Wang, D.H.; Wang, Q.F. Online and offline handwritten Chinese character recognition: Benchmarking on new datasets. *Pattern Recognit.* **2013**, *46*, 155–162. [CrossRef]
42. Boufenar, C.; Kerboua, A.; Batouche, M. Investigation on deep learning for off-line handwritten Arabic character recognition. *Cogn. Syst. Res.* **2018**, *50*, 180–195. [CrossRef]
43. Husnain, M.; Missen, M.M.S.; Mumtaz, S.; Jhanidr, M.Z.; Coustaty, M.; Luqman, M.M.; Ogier, J.M.; Choi, G.S. Recognition of urdu handwritten characters using convolutional neural network. *Appl. Sci.* **2019**, *9*, 2758. [CrossRef]
44. Ahmed, S.B.; Naz, S.; Swati, S.; Razzak, M.I. Handwritten Urdu character recognition using one-dimensional BLSTM classifier. *Neural Comput. Appl.* **2019**, *31*, 1143–1151. [CrossRef]
45. Kavitha, B.R.; Srimathi, C. Benchmarking on offline Handwritten Tamil Character Recognition using convolutional neural networks. *J. King Saud Univ.-Comput. Inf. Sci.* **2019**, *34*, 1183–1190. [CrossRef]
46. Dewan, S.; Chakravarthy, S. A system for offline character recognition using auto-encoder networks. In Proceedings of the the International Conference on Neural Information Processing, Doha, Qatar, 12–15 November 2012.
47. Sarkhel, R.; Das, N.; Das, A.; Kundu, M.; Nasipuri, M. A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular indic scripts. *Pattern Recognit.* **2017**, *71*, 78–93. [CrossRef]
48. Gupta, A.; Sarkhel, R.; Das, N.; Kundu, M. Multiobjective optimization for recognition of isolated handwritten Indic scripts. *Pattern Recognit. Lett.* **2019**, *128*, 318–325. [CrossRef]
49. Nguyen, C.T.; Khuong, V.T.M.; Nguyen, H.T.; Nakagawa, M. CNN based spatial classification features for clustering offline handwritten mathematical expressions. *Pattern Recognit. Lett.* **2020**, *131*, 113–120. [CrossRef]
50. Ziran, Z.; Pic, X.; Undri Innocenti, S.; Mugnai, D.; Marinai, S. Text alignment in early printed books combining deep learning and dynamic programming. *Pattern Recognit. Lett.* **2020**, *133*, 109–115. [CrossRef]
51. Ptucha, R.; Petroski Such, F.; Pillai, S.; Brockler, F.; Singh, V.; Hutkowski, P. Intelligent character recognition using fully convolutional neural networks. *Pattern Recognit.* **2019**, *88*, 604–613. [CrossRef]
52. Tso, W.W.; Burnak, B.; Pistikopoulos, E.N. HY-POP: Hyperparameter optimization of machine learning models through parametric programming. *Comput. Chem. Eng.* **2020**, *139*, 106902. [CrossRef]
53. Cui, H.; Bai, J. A new hyperparameters optimization method for convolutional neural networks. *Pattern Recognit. Lett.* **2019**, *125*, 828–834. [CrossRef]
54. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
55. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
56. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
57. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
58. Ghosh, T.; Abedin, M.H.Z.; Al Banna, H.; Mumenin, N.; Abu Yousuf, M. Performance Analysis of State of the Art Convolutional Neural Network Architectures in Bangla Handwritten Character Recognition. *Pattern Recognit. Image Anal.* **2021**, *31*, 60–71. [CrossRef]

59. LeCun, Y. The Mnist Dataset of Handwritten Digits. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 26 February 2022).
60. Kaggle:A-Z Handwritten Alphabets in.csv Format. Available online: https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format/metadata (accessed on 26 February 2022).
61. Kavitha, M.; Gayathri, R.; Polat, K.; Alhudhaif, A.; Alenezi, F. Performance evaluation of deep e-CNN with integrated spatial-spectral features in hyperspectral image classification. *Measurement* **2022**, *191*, 110760. [CrossRef]
62. Foysal Haque, K.; Farhan Haque, F.; Gandy, L.; Abdelgawad, A. Automatic Detection of COVID-19 from Chest X-ray Images with Convolutional Neural Networks. In Proceedings of the 2020 International Conference on Computing, Electronics and Communications Engineering (ICCECE), Southend Campus, UK, 17–18 August 2020; pp. 125–130.
63. Mor, S.S.; Solanki, S.; Gupta, S.; Dhingra, S.; Jain, M.; Saxena, R. Handwritten text recognition: With deep learning and android. *Int. J. Eng. Adv. Technol.* **2019**, *8*, 172–178.
64. Alom, M.Z.; Sidike, P.; Taha, T.M.; Asari, V.K. Handwritten Bangla Digit Recognition Using Deep Learning. *arXiv* **2017**, arXiv:1705.02680.
65. Sabour, S.; Frosst, N.; Hinton, G.E. Dynamic routing between capsules. In Proceedings of the 2007 Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017; pp. 548–556.
66. Dos Santos, M.M.; da Silva Filho, A.G.; dos Santos, W.P. Deep convolutional extreme learning machines: Filters combination and error model validation. *Neurocomputing* **2019**, *329*, 359–369. [CrossRef]
67. Adnan, M.; Rahman, F.; Imrul, M.; AL, N.; Shabnam, S. Handwritten Bangla Character Recognition using Inception Convolutional Neural Network. *Int. J. Comput. Appl.* **2018**, *181*, 48–59. [CrossRef]
68. Xue, W.; Dai, X.; Liu, L. Remote Sensing Scene Classification Based on Multi-Structure Deep Features Fusion. *IEEE Access* **2020**, *8*, 28746–28755. [CrossRef]
69. Prashanth, D.S.; Mehta, R.V.K.; Sharma, N. Classification of Handwritten Devanagari Number-An analysis of Pattern Recognition Tool using Neural Network and CNN. In *Procedia Computer Science*; Elsevier: Amsterdam, The Netherlands, 2020; Volume 167, pp. 2445–2457.
70. Joshi, D.S.; Risodkar, Y.R. Deep Learning Based Gujarati Handwritten Character Recognition. In Proceedings of the 2018 International Conference On Advances in Communication and Computing Technology, Sangamner, India, 8–9 February 2018; pp. 563–566.
71. Sen, S.; Shaoo, D.; Paul, S.; Sarkar, R.; Roy, K. Online handwritten bangla character recognition using CNN: A deep learning approach. In *Advances in Intelligent Systems and Computing*; Springer: Singapore, 2018; Volume 695, pp. 413–420. ISBN 9789811075650.
72. Weng, Y.; Xia, C. A New Deep Learning-Based Handwritten Character Recognition System on Mobile Computing Devices. *Mob. Netw. Appl.* **2020**, *25*, 402–411. [CrossRef]
73. Gan, J.; Wang, W.; Lu, K. A new perspective: Recognizing online handwritten Chinese characters via 1-dimensional CNN. *Inf. Sci. (Ny).* **2019**, *478*, 375–390. [CrossRef]
74. Saha, S.; Saha, N. A Lightning fast approach to classify Bangla Handwritten Characters and Numerals using newly structured Deep Neural Network. *Procedia Comput. Sci.* **2018**, *132*, 1760–1770. [CrossRef]
75. Hamdan, Y.B. Sathish Construction of Statistical SVM based Recognition Model for Handwritten Character Recognition. *J. Inf. Technol. Digit. World* **2021**, *3*, 92–107. [CrossRef]
76. Ukil, S.; Ghosh, S.; Obaidullah, S.M.; Santosh, K.C.; Roy, K.; Das, N. Improved word-level handwritten Indic script identification by integrating small convolutional neural networks. *Neural Comput. Appl.* **2020**, *32*, 2829–2844. [CrossRef]
77. Cavalin, P.; Oliveira, L. Confusion matrix-based building of hierarchical classification. In Proceedings of the Pattern Recognition, Image Analysis, Computer Vision, and Applications, Havana, Cuba, 28–31 October 2019; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11401, pp. 271–278.